Terbit *online* pada laman : http://teknosi.fti.unand.ac.id/

Jurnal Nasional Teknologi dan Sistem Informasi

| ISSN (Print) 2460-3465 | ISSN (Online) 2476-8812 |

Ekstraksi Basis Pengetahuan Ke Dalam Basisdata Graf Menggunakan Graf Properti

Wahyudi[1,2] , *Fajril Akbar*[1]

[1]*Jurusan Sistem Informasi Fakultas Teknologi Informasi Universitas Andalas Padang, Indonesia*
[2]*Sekolah Teknik Elektro dan Informatika ITB Bandung, Indonesia*

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi: 01 Maret 2019
Revisi Akhir: 21 April 2019
Diterbitkan *Online*: 30 April 2019

KATA KUNCI

Basis pengetahuan,
Graf, property,
Yago

KORESPONDENSI

Phone: (0751)-9824667

E-mail: wahyudi@fti.unand.ac.id

A B S T R A C T

Salah satu jenis commonsense knowledge adalah basis pengetahuan, yaitu kumpulan fakta dan informasi umum yang diketahui manusia. Basis pengetahuan yang tersedia sangat banyak, kami memilih Yago karena Yago menggabungkan beberapa sumber data menjadi sebuah ontologi. Fakta fakta ini saling terhubung sehingga representasi data graf adalah representasi data yang paling tepat untuk ini. Kita perlu melakukan ekstraksi basis pengetahuan kedalam basisdata graf supaya bisa digunakan untuk berbagai macam aplikasi lainnya. Tools basisdata graf Neo4j dipilih karena bisa melakukan penyimpanan dan pemrosesan graf dan bersifat opensource. Algoritma GPE digunakan untuk melakukan ekstraksi basis pengetahuan kedalam basisdata graf. Penelitian yang kami lakukan menghasilkan sebuah basisdata graf dengan jumlah *node* : 6.519.734 dan *edge* yang dihasilkan 18.724.395. Pengujian menggunakan *query* dengan berbagai permasalahan graf berhasil dilakukan dan memberikan hasil yang diharapkan

1. PENDAHULUAN

Graf merupakan cabang ilmu matematika yang berkaitan dengan disiplin ilmu lainnya. Teori graf ini pertama kali dikenalkan pada makalah "Seven Bridges of Königsberg" yang ditulis oleh Leonhard Euler pada tahun 1736 [1]. Di matematika dan ilmu komputer, teori graf adalah pokok bahasan yang mempelajari tentang graf, suatu struktur matematis yang biasa digunakan untuk memodelkan sekumpulan objek dan relasi atau hubungan di antara objek-objek tersebut.

Fleksibelitas model graf memungkinkan kita untuk menambahkan entitas dan relasinya tanpa mempengaruhi atau merubah data yang sudah ada [1]. Beberapa aplikasi web terkenal seperti facebook[2], google[3], Wikipedia dan Imdb serta banyak aplikasi lainnya menggunakan representasi graf sebagai representasi data[4].

Representasi graf memungkinkan pertambahan data menjadi lebih mudah. Data yang mempunyai struktur berbeda bisa ditambahkan secara langsung tanpa perlu merubah basisdata yang ada[5]. Pertumbuhan data yang semakin besar memunculkan sebuah istilah yang disebut data besar. Data besar merupakan sebuah istilah yang sangat populer saat ini. Data besar merupakan himpunan data atau dataset dalam jumlah yang sangat besar dan kompleks sehingga menjadikannya sukar ditangani apabila hanya menggunakan perkakas manajemen basisdata biasa atau aplikasi pemroses data tradisional saja[6]. Lebih spesifik lagi, data besar memiliki karakteristik dengan 4 V yaitu : *Volume* untuk ukuran data, *Velocity* (kecepatan) untuk streaming dan dinamisasi data, *Variety* untuk bentuk data yang berbeda dan *Veracity* untuk data yang tidak jelas atau data yang berkualitas jelek[7]. Data besar berasal dari data jejaring sosial (seperti facebook, twitter, weibo dll), sistem *e-coomerce* (seperti amazon, lazada), keuangan (transkasi saham), sensor networks, *e-government*, penelitian ilmiah dan *semantic knowledge bases*.

Penelitian Ramanujam [8] melakukan ekstraksi data semi terstruktur yaitu *rdf* (*Resource Description framework*) ke dalam basisdata, yaitu basisdata relasional. Penelitian ini menggunakan proses *mapping schema* dan proses transformasi ke basisdata relasional. Penelitian ini menunjukkan begitu kompleks dan rumitnya membuat basisdata relasional dari data *rdf*. Penelitian yang kami lakukan adalah ekstraksi data *rdf* ke dalam basisdata graf. Ekstraksi data ke dalam basisdata graf lebih praktis dan simpel dibandingkan dengan basisdata relasional[9]. Penelitian Angles[9] mengusulkan basisdata graf untuk mengatasi kerumitan dan keterbatasan pada basisdata relasional. Sembilan basisdata graf dibandingkan dalam penelitian ini, salah satunya adalah Neo4j. Penelitian yang kami lakukan sama dengan [8],

https://doi.org/10.25077/ TEKNOSI.v5i1.2019.41-48

perbedaannya adalah kami menggunakan basisdata graf untuk menyimpan hasil ekstraksi *rdf*.

Knowledge base atau dalam makalah ini disebut basis pengetahuan menyediakan banyak informasi tentang berbagai macam entitas seperti : orang, sungai, gedung, negara dan lain sebagainya. Basis pengetahuan juga memiliki fakta yang menghubungkan beberapa entitas tersebut, seperti : siapa dilahirkan dimana, sungai apa yang terdapat di suatu kota, siapa penyanyi yang menyanyikan lagu tersebut, dan banyak lainnya[10]. Fakta yang terdapat pada basis pengetahuan berasal dari halaman Wikipedia dan website lainnya seperti *Geonames* dan lain lain. Fakta ini tidak bisa langsung digunakan karena masih bercampur dengan data lainnya yang tidak diperlukan. Dalam penelitian ini kami melakukan ekstraksi fakta tersebut dan menyimpannya kedalam basisdata graf sehingga bisa digunakan untuk berbagai macam disiplin ilmu seperti : sistem temu kembali informasi, sistem tanya jawab, sistem cerdas dan lain sebagainya.

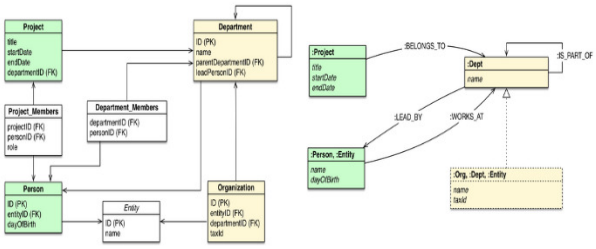
Basis pengetahuan merupakan informasi yang dieksrak dari web seperti Wikipedia dan disimpan dalam format *triple rdf* seperti Yago[11], DBPedia[12]. Dalam penelitian ini kami menggunakan basis pengetahuan Yago. Sistematika penulisan pada makalah ini adalah sebagai berikut : pendahuluan berupa latar belakang dan masalah penelitian. Bab berikutnya kami menjelaskan metodologi penelitian, pada bab ini akan dijelaskan alasan kenapa memilih representasi data graf. Kami juga menjelaskan alasan memilih Yago dan Neo4j sebagai *tools* basisdata. Bab berikutnya kami menjelaskan hasil penelitian dan pembahasan serta bab terakhir berisi kesimpulan dari makalah ini

1.1. Ekstraksi Data rdf

Penelitian ekstraksi data *rdf* dilakukan oleh Pan[13] dan Ramanujam[8]. Penelitian tersebut melakukan ekstraksi data *rdf* ke dalam basisdata relasional. Hybrid model[13] digunakan untuk melakukan ekstraksi data dari *rdf*. Pendekatan ini menggunakan gabungan pendekatan tabel properti dengan kelas horizontal. Basisdata relasional yang digunakan adalah MS Access. Kelemahan penelitian ini adalah menghasilkan banyak tabel yang tidak perlu untuk memetakan 1: N relasi antara subjek dan objek pada data *rdf*. Penyempurnaan penelitian ini dilakukan dengan metode R2D[8], dengan cara membuat *foreign key* untuk mengurangi banyak tabel yang tidak dibutuhkan. Tetapi penelitian ini juga memiliki kelemahan yaitu tabel yang dihasilkan masih banyak, karena setiap entitas memiliki atribut yang berbeda beda sehingga membutuhkan tabel yang berbeda juga. Semakin banyak entitas semakin banyak tabel yang dibutuhkan (lihat bagian 2.2). Pendekatan basisdata graf [9],[14],[15] mengatasi hal tersebut, dengan *direct pointer* pada representasi graf keterhubungan antara subjek-relasi-objek bisa langsung dilakukan tanpa tabel.

1.2. Basisdata Graf

Kelebihan model graf dalam representasi hubungan antar objek atau *node* mulai diadopsi ke dalam basidata. Hal ini bermula ketika para developer aplikasi menemukan kerumitan dan lamanya proses untuk melakukan *query* pada *big data*. Mereka mulai berfikir tentang kemungkinan menggantikan tabular SQL dengan model graf. Mereka melihat bahwa graf jauh lebih mudah digunakan developer yang bekerja pada data terhubung. Dan hal ini sebenarnya bukan pendekatan baru, karena teori graf adalah teori lama dan sudah dikembangkan di berbagai bidang[1]. Salah satu pengembangan basisdata graf adalah pemodelan jejaring sosial[15]. Facebook, google dan twitter adalah beberapa perusahaan yang menggunakan basisdata graf[1].



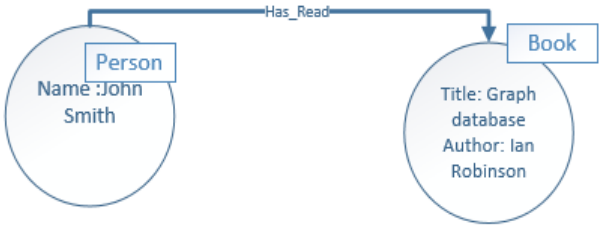
Gambar.2 Basisdata relasional vs Basisdata graf

Secara umum basisdata terbagi menjadi 2 yaitu: 1. Basisdata relasional atau RDBMS (relational database management system) dengan My SQL atau MS SQL yang sangat baik untuk pengelolaan data keuangan atau data organisasi yang terstruktur. 2. Basisdata graf , yang sangat baik untuk pengelolaan basisdata yang tidak terstruktur. Seperti catatan pengalaman harian dan pengetahuan umum yang terus berubah. Berbeda dengan konsep RDBMS yang berdasar pada tabel dan kata kunci, basisdata graf berdasar pada hubungan antara *Node*, yang secara ringkas seperti penampakan Gambar 2 di atas.

Dari Gambar 2 bisa disimpulkan bahwa transaksi pada basisdata relasional dikendalikan dengan PK (primary key) dan FK (foreign key), dengan data tersimpan dalam tabel-tabel terstruktur yang cukup banyak. Pada gambar di atas ada 4 PK dan 8 FK yang disimpan dalam 4 tabel dan 3 tabel sementara. Untuk setiap transaksi, seluruh PK dan FK harus selalu dicatat dalam memori, sehingga setiap transaksi akan memakai ukuran RAM yang cukup besar. Pendekatan berbeda dilakukan oleh Basisdata graf, yaitu hanya memerlukan 3 *node* dan sejumlah keterangan relationship, untuk suatu permintaan transaksi data, kita dapat fokus pada satu relationship saja, sehingga akan menghemat ukuran RAM yang digunakan.

Dalam penelitian ini model struktur graf yang digunakan adalah model graf properti. Model graf properti adalah sebuah graf yang terdiri dari *node, edge* dan properti. Properti yang digunakan dalam penelitian ini adalah properti nn (*node name*) pada *node* dan *rdf_type* pada *edge*. Pada gambar 3 terdapat dua buah *node* yaitu *node* dengan label *person* dan *node* dengan label *book* dan sebuah *edge* yaitu *hasRead*. *Node person* mempunyai sebuah properti yaitu *name* dengan nilai John Smith, sedangkan *node* book memiliki 2 buah properti yaitu title dan author. Title memiliki nilai graph database dan author memiliki nilai ian robinson[16].

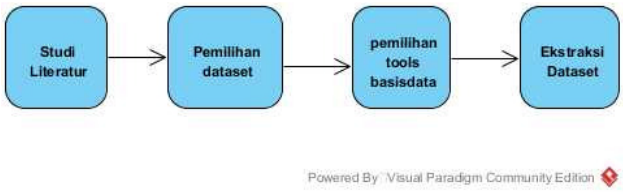
 https://doi.org/10.25077/ TEKNOSI.v5i1.2019.41-48



Gambar 3. Graf properti[16]

2. METODE

Metodologi penelitian yang digunakan adalah seperti pada gambar 4 .



Gambar 4 . Metodologi penelitian

2.1. *Pemilihan dataset*

Yago dikenalkan oleh Suchanek [10], merupakan suatu teknik yang digunakan untuk ekstraksi data yang terdapat pada Wikipedia dan menggunakan ontology Wordnet. Yago bukan hanya sekedar ekstraksi data, tetapi juga mengembangkan ontologi yang tidak terdapat di Wordnet, sebagaimana kita ketahui bahwa banyak informasi pada Wikipedia yang belum memiliki kelas atau hirarki yang menggambarkan konsep taksonomi, sedangkan Wordnet memiliki hirarki yang jelas tetapi tidak menampung semua hirarki yang terdapat di Wikipedia. Kekosongan inilah yang coba diisi oleh Yago dengan mengembangkan sebuah ontologi baru berbasiskan Wordnet pada Wikipedia. Pendekatan Yago dengan Wikipedia menggunakan infoboxes Wikipedia dan halaman kategori, infoboxes jauh lebih mudah untuk mengexploitasi dan mengurai teks. Halaman kategori mengelompakkan sebuah artikel termasuk kategori tertentu, contohnya: elvis termasuk kategori penyanyi rock amerika. Daftar ini memberikan kita kandidat untuk entitas (contoh: elvis), calon konsep (contoh: Isa(Elvis,rockSinger)). Dalam ontologi, konsep harus diatur dalam sebuah taksonomi. Kategori Wikipedia memang diatur dalam sebuah hirarki, tetapi hirarki ini tidak sama dengan tujuan ontologi. Contohnya : Elvis masuk kategori Grammy award pada Wikipedia, tetapi elvis adalah pemenang grammy award bukan kategori grammy award.

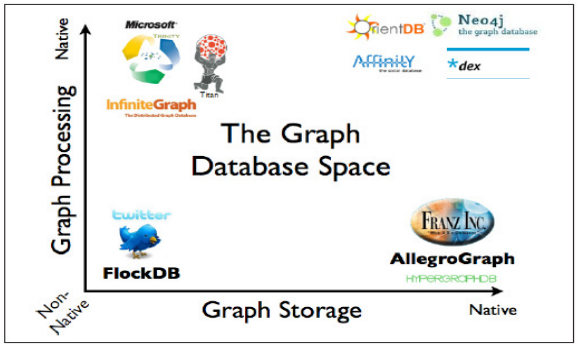
Yago menyajikan teknik yang mempunyai akurasi yang tinggi. Dengan menggabungkan sumber dari fakta yang terdapat pada Wikipedia dan Wordnet. Hal ini memungkinkan Yago mempunyai keuntangan sendiri. Di satu sisi dia mengambil entitas dan fakta yang terdapat di Wikipedia, di sisi lain menggunakan taksonomi yang lengkap dan bersih dari Wordnet[13]. Yago mempunyai lebih dari 1,7 juta entitas dan 15 juta fakta dari Wikipedia, Geonames dan Wordnet dengan akurasi di atas 95 %[11][17], seperti pada tabel 1.

Tabel 1 Akurasi Yago,[11]

Relation	# evaluated facts	Accuracy
subClassOf	298	97.70% ± 1.59%
Type	343	94.54% ± 2.36%
familyNameOf	221	97.81% ± 1.75%
givenNameOf	161	97.62% ± 2.08%
establishedIn	170	90.84% ± 4.28%
bornInYear	170	93.14% ± 3.71%
diedInYear	147	98.72% ± 1.30%
locatedIn	180	98.41% ± 1.52%
politicianOf	176	92.43% ± 3.93%
writtenInYear	172	94.35% ± 3.33%
hasWonPrize	122	98.47% ± 1.57%

2.2. *Pemilihan tools basisdata*

Basisdata graf merupakan basisdata yang menggunakan struktur graf (*node, edge*, properti) untuk menyimpan data. Basisdata graf menyediakan *index-free adjacency* yang artinya setiap elemen berisi *direct pointer* ke *adjacent element* dan tidak membutuhkan lagi suatu *index lookups*. Beberapa basisdata graf menggunakan penyimpanan graf *native*, yang dioptimalkan dan dirancang untuk menyimpan dan mengelola graf. Tidak semua teknologi basisdata graf menggunakan penyimpanan graf *native*, Ada juga basisdata graf yang disimpan ke dalam basisdata relasional, berorientasi objek, atau jenis lain dari penyimpanan NoSQL[1][14]. Sekarang ini tools basidata graf sudah tergabung dengan pemrosesan graf. Hal ini memudahkan kita untuk dapat menyelasaikan permasalahan graf dengan menggabungkan pendekatan pengambilan *query* dan pemrosesan graf yang kompleks dalam sebuah *tools* basisdata graf



Gambar 5. Overview jenis-jenis basisdata graf, [1]

Gambar 5 menjelaskan perbandingan beberapa basisdata graf. Ada 2 hal yang dibandingkan, yaitu : penyimpanan graf dan pemrosesan graf. Parameternya adalah *native* dan *non native*. Pada penyimpanan graf, basisdata graf *native* akan menyimpan data dalam bentuk graf (*node, edge*, dan property) sedangkan yang *non native* akan menyimpan data dalam bentuk basisdata relasional, basisdata berorientasi objek, atau jenis lain dari penyimpanan NoSQL. Begitu juga pada pemroesan graf, pemrosesan graf *native* akan memproses graf dalam bentuk *node*, *edge* begitu juga sebaliknya. Hal ini memudahkan pengembang untuk dapat menyelasaikan permasalahan pada graf dengan

menggabungkan pendekatan pengambilan *query* dan pemrosesan graf yang kompleks. Neo4j dipilih karena memiliki kelebihan dibandingkan dengan basisdata graf lainnya yaitu penyimpanan graf dan pemrosesan graf nya dalam bentuk *native.* Neo4j melakukan pemrosesan graf tanpa perlu melakukan perubahan pada penyimpanan datanya, karena kedua nya menggunakan graf *native*, begitu juga sebaliknya.

2.3. Ekstraksi dataset

Dalam penelitian ini, digunakan fakta Yago yang terdapat dalam 3 buah file Yago seperti pada tabel 2. File Yago tersebut adalah *Yago facts*, *Yago date facts*, dan *Yago literal facts*. Untuk melakukan ekstraksi, maka kami menggunakan algoritma ekstraksi yang kami sebut *graph property extraction* (GPE). Langkah pertama yang dilakukan adalah membersihkan file Yago di atas sehingga menjadi bentuk *triple rdf*.

Tabel 2. file Yago yang digunakan

No	File Name	*Node*	*Edge*
1	YagoFacts	3.43 M	5.62 M
2	YagoLiteralFacts	2.57 M	1.61 M
3	YagoDatefact	2.32 M	3.18 M

M = Million

Langkah pertama yang dilakukan adalah membersihkan file Yago di atas sehingga menjadi bentuk *triple rdf*.. tahapan ini bisa dilihat pada gambar 6 yaitu file *triple rdf* yang belum dibersihkan

```
        <yagoTheme_yagoFacts>   <hasGloss>      "This file is part of the ontology YAGO3. It is licensed under a Cre
<id_rB6isMnplh_H?S_LT?Qo1Fzc!>  <Jesús_Rivera_Sánchez>  <isLeaderOf>    <Pueblo_of_Naranjito>  ↓
<id_BOK!FvTDPu_H?S_1NVSTKkFbS>  <Elizabeth_II>  <isLeaderOf>    <Royal_Numismatic_Society>      ↓
<id_Uy5EwU3nX1_H?S_otuJrkvKs1>  <Richard_Stallman>      <isLeaderOf>    <Free_Software_Foundation>      ↓
<id_A?rIHtKpyX_H?S_Ap3TBzfE6b>  <Keith_Peterson>        <isLeaderOf>    <Cambridge_Bay> ↓
<id_vzhzgmCR5Y_H?S_9xW07qYiaH>  <William_H._Seward_Jr.> <isLeaderOf>    <9th_New_York_Heavy_Artillery_Regiment> ↓
<id_GqUh9jFAN?_H?S_A39fu5FWu4>  <Andranik>      <isLeaderOf>    <Armenian_fedayi>       ↓
<id_s60Psk1DHb_H?S_OACCn8W8Kv>  <Ramasamy_Palanisamy>   <isLeaderOf>    <Democratic_Action_Party_(Malaysia)>    ↓
<id_pII60Mnz8o_H?S_8mvRWxKXDG>  <Matt_Bevin>    <isLeaderOf>    <Kentucky_Air_National_Guard>   ↓
<id_losV58WRWE_H?S_KxmZk2LtbV>  <Leonard_Leo>   <isLeaderOf>    <Federalist_Society>    ↓
<id_b2c!d9tP5s_H?S_G1fqe41nrx>  <James_Vincent_Cleary>  <isLeaderOf>    <Bennington,_New_Hampshire>     ↓
<id_SuFKiU!7li_H?S_Lade3HulqA>  <Amjad_Bashir>  <isLeaderOf>    <Khushab_District>      ↓
```

Gambar 6. *Triple rdf* Yago facts

Di dalam file tersebut kita bisa melihat pada baris pertama tentang informasi file, baris kedua dan seterusnya berisi tentang isi dari file *YagoFacts*, berupa id dan *triple rdf*. File ini yang akan diedit dan diubah formatnya dari *tsv* ke *csv* sebelum dikestrak ke Neo4j.

Yang perlu diedit adalah menghapus id dan karakter <> di setiap *triple rdf* karena tidak penting dan menambah header file berupa subjek, relasi dan objek. Hasilnya bisa dilihat pada gambar 7.

```
"subjek"                        "relasi"        "objek"↓
"Jesús_Rivera_Sánchez"          "isLeaderOf"    "Pueblo_of_Naranjito"↓
"Elizabeth_II"                  "isLeaderOf"    "Royal_Numismatic_Society"↓
"Richard_Stallman"              "isLeaderOf"    "Free_Software_Foundation"↓
"Keith_Peterson"                "isLeaderOf"    "Cambridge_Bay"↓
"William_H_Seward_Jr"           "isLeaderOf"    "9th_New_York_Heavy_Artillery_Regiment"↓
"Andranik"                      "isLeaderOf"    "Armenian_fedayi"↓
"Ramasamy_Palanisamy"           "isLeaderOf"    "Democratic_Action_Party_(Malaysia)"↓
"Matt_Bevin"                    "isLeaderOf"    "Kentucky_Air_National_Guard"↓
"Leonard_Leo"                   "isLeaderOf"    "Federalist_Society"↓
"James_Vincent_Cleary"          "isLeaderOf"    "Bennington,_New_Hampshire"↓
"Amjad_Bashir"                  "isLeaderOf"    "Khushab_District"↓
"Tushar_Amarsinh_Chaudhary"     "isLeaderOf"    "Mota,_Gujarat"↓
```

Gambar 7. *Triple rdf* Yago facts yang sudah di bersihkan

Algoritma GPE yang kami buat mempunyai input file *triple rdf* yang terdiri dari subjek, relasi dan objek yang dinotasikan sebagai H(sub,rel,obj). File *triple rdf* ini akan diekstraksi menjadi basisdata graf dalam bentuk *node* dan *edge* dinotasikan G(V,E). Langkah pertama adalah inisialisasi *node* dan *edge*. Kemudian langkah selanjutnya adalah mengubah setiap baris *triple rdf* (sub,rel, obj) menjadi bentuk graf. Ada tiga bentuk ekstraksi yang dilakukan, yang pertama untuk setiap baris *triple rdf* H yang belum mempunyai *node* dan *edge* di basisdata graf, maka dibuat *node* (V) dari subjek (sub) dan objek (obj), kemudian baru dibuat *edge* yang menghubungkan kedua *node* tersebut dari relasi (rel). Yang kedua, jika salah satu sub atau obj yang sudah terdapat pada basisdatagraf, maka *node* yang kita buat adalah *node* yang belum ada dibasisdata graf, sedangkan *node* yang sudah ada tidak perlu dibuat lagi, kemudian kita hubungkan dengan rel pada *triple rdf* tersebut. Bentuk ketiga adalah untuk *node* yang sudah ada dari sub dan obj, maka kita hanya membuat rel yang menghubungkan

https://doi.org/10.25077/ TEKNOSI.v5i1.2019.41-48

antara sub dan obj pada *triple rdf*. Untuk lebih lengkapnya algoritma GPE bisa dilihat pada gambar 8 .

3. HASIL DAN PEMBAHASAN

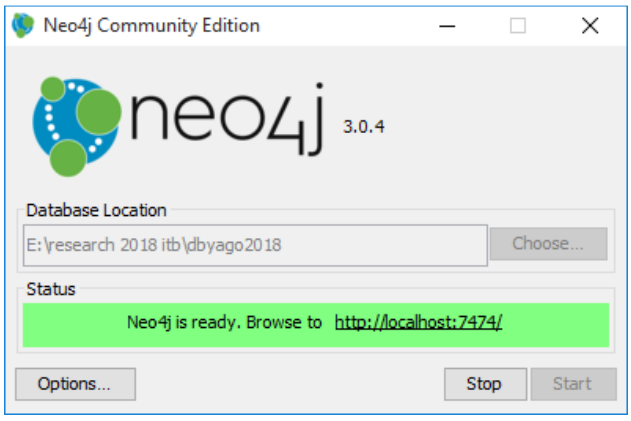
Pada bagian ini kami menjelaskan hasil penelitian dan pembahasanannya. Proses ekstraksi basis pengetahuan Yago menggunakan algoritma GPE menghasilkan sebuah basisdata graf yang menghasilkan *node* sebanyak : 6.519.734 dan *edge* yang dihasilkan 18.724.395. Basisdata graf yang kami gunakan adalah Neo4j versi 3.0.4, tampilannya bisa dilihat pada gambar 9. Spesifikasi hardware yang kami gunakan untuk melakukan pengujian ini adalah laptop dell inspiron 3442 dengan spesifikasi Prosesor Intel Core i3 4005U-1.7Ghz, 8 GB RAM, GPU Intel HD 4000 Graphics.

```
Input: File triple rdf H(sub,rel,obj)

Output: a graph G (V, E)

1. set V= Ø, E=Ø
2. foreach  (sub,rel,obj)ϵ H(i) do
      a. if (u,v) ϵ V = Ø
            i.  create a node v ϵ V where v(i) = sub
           ii.  create a node u ϵ V where u(i) = obj
          iii.  create an edge e ϵ E where e(u(i),v(i))= rel
      b. else if (u|v) ϵ V = Ø
            i.  create a node u|v ϵ V where u(i)|v(i) = sub|obj
           ii.  create an edge e ϵ E where e(u(i),v(i))= rel
      c. else
            i.  create an edge e ϵ E where e(u(i),v(i))= rel
3. return G(V,E)
```

Gambar 8 Algoritma *Graph Property Extraction (GPE)*



Gambar 9. Tampilan awal Neo4j

Neo4j sudah menyediakan aplikasi berbasis web untuk dapat mengakses databasenya. Gunakan browser dan masuk ke dalam lokal server Neo4j dengan alamat http://localhost:7474/. Untuk penulisan *query* pada Neo4j menggunakan bahasa *cypher*. *cypher* khusus untuk penulisan *query* pada graph database. Pada graph ada 2 elemen penting, yaitu *node* dan *edge*. Kita menyimpan data dalam bentuk *node* dan *edge* dengan menggunakan *cypher*. Istilah pada *cypher* selain *node* dan *edge* juga ada istilah *pattern* atau pola *cypher*. C*ypher* juga mengenal istilah basis data seperti DDL dan DML.

Secara umum struktur *cypher* hampir sama dengan sql, misalnya untuk membaca data dari graph maka *cypher* menggunakan clause "MATCH" dan "RETURN" untuk mendapatkan nilai / variabel yang diinginkan. Contoh nya : untuk *query*

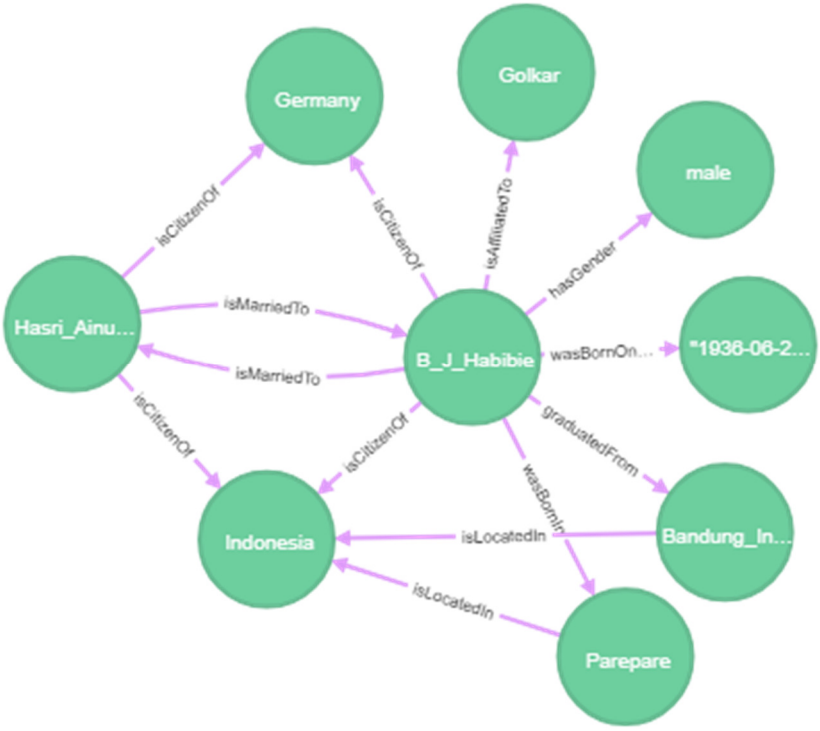
```
match(n:owl_Thing{nn:'B_J_Habibie'})-[r]-
(o) return n,r,o
```

ketika *query* ini dieksekusi akan menampilkan seluruh *node* dan *edge* yang terhubung dengan *node* B.J Habibie sejumlah 8 *node* dan 9 relasi, seperti pada gambar 10.

Selain menampilkan banyak *node*, kita juga bisa menampilkan *query* yang spesifik untuk satu relasi, pada misalnya kita ingin mengetahui di kota manakah bj habibie dilahirkan. Dalam masalah ini jika dimodelkan kedalam bentuk graf adalah sebuah *node* terhubung dengan relasi kota tempat lahir, yang dicari adalah *node* yang terhubung dengan relasi tersebut. Di dalam basis data, properti *node* nya adalah nn, datanya adalah B.J.Habibie terhubung dengan sebuah relasi yang mempunyai property *rdf_type wasBornIn*, yang artinya tempat lahir, secara lengkap querynya adalah sebagai berikut :

```
match(n:owl_Thing{nn:'B_J_Habibie'})-
[r{rdf_type:'wasBornIn'}]-(o) return o
```

Hasil *query* tersebut adalah sebuah *node* yang bernama parepare, dapat dilihat pada tabel 3.



Gambar 10. Hasil eksekusi *query* pada Neo4j

. Contoh berikutnya adalah kita ingin mengetahui hubungan B.J.habibie dengan Indonesia, masalah ini jika dimodelkan dalam bentuk graf adalah dua *node* yaitu B.J.Habibie dan Indonesia, yang kita cari adalah relasi yang menghubungkan dua *node* tersebut. *Cypher* Neo4j digunakan untuk mendapatkan relasi antara dua *node* tersebut

```
MATCH(n:owl_Thing{nn:'B_J_Habibie'})-[r]-
>(o:owl_Thing{nn:'Indonesia'}) RETURN r
```

Hasilnya adalah sebuah relasi *isCitizenOf*, seperti pada tabel 3.

Tabel 3. Beberapa permasalahan yang diselesaikan dengan basisdata graf

No	Permasalahan	Bentuk graf			Query	Hasil
1	Diketahui sebuah *node* dan *edge*, dicari *node* yang berhubungan dengan *edge* tersebut. Contoh : di kota manakah bj habibie dilahirkan	nn:B_J_ Habibie	wasBornIn	??	`match(n:owl_Thing{nn:'B_J_ Habibie'})-[r{rdf_type:'wasBornIn'}]-(o) return o`	Parepare
2	Diketahui dua buah *node*, dicari *edge* yang terhubung dengan dua *node* tersebut. Contoh: Apa hubungan B.J.habibie dengan Indonesia	nn:B_J_ Habibie	???	nn:Indonesia	`MATCH (n:owl_Thing{nn:'B_J_Habibie'})-[r]->(o:owl_Thing{nn:'Indonesia'}) RETURN r`	rdf_type isCitizenOf

Kami juga melakukan pengujian dengan membandingkan eksekusi *query* dengan *runing time query Neo4j*. Penelitian kami menggunakan *index node* dengan properti nn untuk meningkatkan waktu eksekusi query. Index berguna untuk mempercepat pencarian data dengan menggunakan properti tertentu. Pengujian dilakukan pada web server Neo4j. Pengujian ini menampilkan visualisasi graf, sehingga membutuhkan waktu tambahan dibandingkan pada pemrosesan graf. Untuk eksekusi *query* yang menghasilkan 10 – 100 buah *node* memerlukan running time 2- 5 detik. Hasil lengkapnya terdapat pada tabel 4

 https://doi.org/10.25077/ TEKNOSI.v5i1.2019.41-48

Tabel 4. Perbandingan waktu eksekusi Neo4j

No	Waktu eksekusi (detik)	Jumlah *node* yang dihasilkan
1	< 1	< 10
2	2 – 5	10 –100
3	6 - 12	101– 500
4	13 - 20	500 – 800
5	>20	> 800

4. KESIMPULAN

Basis pengetahuan merupakan pengetahuan umum yang diketahui oleh manusia, sumbernya bisa dari internet seperti Wikipedia. Data tersebut diolah oleh berbagai macam metode sehingga menjadi sebuah ontologi dan basis pengetahuan. Basis pengetahuan ini belum bisa sacara langsung digunakan. Penelitian ini bertujuan untuk melakukan filter dan menyimpan basis pengetahuan kedalam basisdata graf. Basisdata graf Neo4j dipilih karena pada Neo4j menggunakan pendekatan dan pemrosesan graf yang *native*. Pemilihan basis pengetahuan Yago karena akurasi yang dimiliki Yago adalah sangat baik, yaitu diatas 95%. Ekstraksi basis pengetahuan ke dalam basisdata graf menggunakan algoritma GPE, hal ini dilakukan karena data yang dimasukkan sangat banyak, untuk menghilangkan redundansi data dan membuat keterhubungan data menjadi lebih mudah untuk didapatkan. Basisdata graf yang dihasilkan setelah diuji dengan beberapa *query* menghasilkan output atau hasil yang diinginkan. Penelitian lanjutan bisa dilakukan dengan mengembangkan berbagai macam aplikasi yang menggunakan basisdata graf ini, seperti temu kembali informasi, sistem tanya jawab dan sistem chatbot serta sistem berbasiskan kecerdasan buatan lainnya.

DAFTAR PUSTAKA

[1] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*, Second edi. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2015.

[2] Z. C. Khan *et al.*, "An Analysis of Facebook's Graph Search," 2014.

[3] X. L. Dong *et al.*, "Knowledge Vault : A Web-Scale Approach to Probabilistic Knowledge Fusion," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pp. 601–610, 2014.

[4] A. Welc *et al.*, "Graph analysis: do we have to reinvent the wheel?," *First International Workshop on Graph Data Management Experiences and Systems*, p. 7:1--7:6, 2013.

[5] A. Kanavos, G. Drakopoulos, and A. Tsakalidis, "Graph Community Discovery Algorithms in Neo4j with a Regularization-based Evaluation Metric," *WEBIST 2017 - Proceedings of the 13th International Conference on Web Information Systems and Technologies*, 2017.

[6] S. Abiteboul, R. Hull, V. Vianu, and F. Databases, *Foundations of databases*, vol. 29, no. 11. United States of America: Addison-Wesley Publishing Company, 1995.

[7] W. Fan and J.-P. Huai, "Querying Big Data: Bridging Theory and Practice," *Journal of Computer Science and Technology*, vol. 29, no. 5, pp. 849–869, 2014.

[8] S. Ramanujam, A. Gupta, L. Khan, S. Seida, and B. Thuraisingham, "R2D: Extracting Relational Structure from RDF Stores," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 2009, pp. 361–366.

[9] R. Angles, "A Comparison of Current Graph Database Models," in *2012 IEEE 28th International Conference on Data Engineering Workshops*, 2012, pp. 171–177.

[10] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A Large Ontology from Wikipedia and WordNet," *Web Semantics*, vol. 6, no. 3, pp. 203–217, 2008.

[11] F. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia," in *Proceedings of the 16th international conference*

on World Wide Web - WWW '07, 2007, p. 697.

[12] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives, "DBpedia : A Nucleus for a Web of Open Data," in *ISWC'07/ASWC'07 Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, 2007, pp. 722–735.

[13] Z. Pan and J. Heflin, "DLDB: Extending Relational Databases to Support Semantic Web Queries," in *Practical and Scalable Semantic Systems*, 2003.

[14] Vojtech Kolomicenko, M. Svoboda, and I. Holubová, "Experimental Comparison of Graph Databases," 2013.

[15] S. Jouili and V. Vansteenberghe, "An empirical comparison of graph databases," pp. 708–715, 2013.

[16] Wahyudi, M. L. Khodra, A. S. Prihatmanto, and C. Machbub, "Knowledge-based graph compression using graph property on Yago," in *2017 3rd International Conference on Science in Information Technology (ICSITech)*, 2017, pp. 127–131.

[17] F. Mahdisoltani, J. Biega, and F. Suchanek, "YAGO3: A Knowledge Base from Multilingual Wikipedias," in *7th Biennial Conference on Innovative Data Systems Research (CIDR 2015)*, 2015, pp. 177–185.

 https://doi.org/10.25077/ TEKNOSI.v5i1.2019.41-48